NASA-TM-111507

Order of Magnitude Reasoning Using Logarithms

P. Pandurang Nayak
RECOM Technologies

Artificial Intelligence Research Branch
NASA Ames Research Center
Mail Stop 269-2
Moffett Field, CA 94035-1000

NASA Ames Research Center

Artificial Intelligence Research Branch

Technical Report FIA-93-26
September, 1993

Order of Magnitude Reasoning using Logarithms

P. Pandurang Nayak
Recom Technologies, NASA Ames Research Center
AI Research Branch
Mail Stop 269-2
Moffett Field, CA 94035.
Email: nayak@ptolemy.arc.nasa.gov

Abstract

Converting complex equations into simpler, more tractable equations usually involves approximation. Approximation is usually done by identifying and removing insignificant terms, while retaining significant ones. The significance of a term can be determined by order of magnitude reasoning. In this paper we describe NAPIER, an implemented order of magnitude reasoning system. NAPIER defines the order of magnitude of a quantity on a logarithmic scale, and uses a set of rules to propagate orders of magnitudes through equations. A novel feature of NAPIER is the way it handles non-linear simultaneous equations, using linear programming in conjunction with backtracking. We show that order of magnitude reasoning in NAPIER is, in general, intractable and then discuss an approximate reasoning technique that allows it to run fast in practice. Some of NAPIER's inference rules are heuristic, and we estimate the error introduced by their use. We also empirically evaluate alternate rules that are guaranteed to be correct.

1 Introduction

Mathematical models are pervasive in science and engineering. Both analytical and numerical techniques have been used to solve the equations resulting from such models. Analytical methods are applicable only to restricted classes of equations (e.g., linear systems). To apply analytical techniques to more complex classes of equations, scientists and engineers have had to find ways of approximating the equations to convert them into a simpler form.

With the advent of fast digital computers, the classes of equations that can be solved by numerical (rather than analytical) methods have grown considerably. However, numerical methods are not a panacea; they have their limitations too. For example, solving systems of non-linear equations can be time consuming, and a good initial guess is required to ensure convergence to a solution. Hence, scientists and engineers still strive to identify appropriate approximations so that the resulting equations are as simple as possible. In addition, numerical methods are sometimes inapplicable. For example, during conceptual design, exact numerical values for exogenous quantities are usually unavailable because most of the details of the design are unspecified. Under such circumstances, an engineer needs to fall back on analytical methods, thereby highlighting the role of appropriate approximations.

The approximation process usually involves identifying and removing insignificant terms, while retaining only the significant ones. Consider the following example, previously discussed in [1; 16], from the domain of acid-base chemistry. An important task in this domain is to find the concentration of H^+ ions in a solution. The concentration of ions in solution depends on the dynamic equilibrium resulting from competing chemical reactions. Consider dissolving an acid, AH, in water. The two reversible reactions that occur, corresponding to the ionization of AH and H_2O, are shown in Figure 1.

$$\begin{aligned} AH &\rightleftharpoons H^+ + A^- \\ H_2O &\rightleftharpoons H^+ + OH^- \end{aligned}$$

Figure 1: Ionization reactions that occur on dissolving AH in water

The equilibrium concentrations of the three ions (H^+, OH^-, A^-) and the acid (AH) are determined by the equations shown in Figure 2. Square brackets denote concentrations; C_a is the initial concentration of the acid; K_w is the ion product of water; and K_a is the ionization constant of the acid.

As has been pointed out in [1; 16], solving this set of equations analytically for $[H^+]$ results in a cubic equation which is difficult to solve. In fact, in problems involving *polyprotic* acids, i.e., acids that can yield more than one H^+ ion, the closed form solution for $[H^+]$ can involve equations of degree five or higher, making the solution significantly harder.

Charge balance:	$[H^+] = [A^-] + [OH^-]$	(1)
Mass balance:	$C_a = [A^-] + [AH]$	(2)
Acid ionization equilibrium:	$K_a[AH] = [A^-][H^+]$	(3)
Water ionization equilibrium:	$K_w = [OH^-][H^+]$	(4)

Figure 2: Equilibrium equations for the ionization reactions.

An alternative to the above approach is to approximate the equations, and hence simplify them. For example, a chemist might guess that the acid is strong, so that $[A^-] \gg [OH^-]$ and $[A^-] \gg [AH]$. This justifies reducing the first equation to $[H^+] = [A^-]$ and the second equation to $C_a = [A^-]$, leading to a straightforward solution.

The reasoning following the assumptions that $[A^-] \gg [OH^-]$ and $[A^-] \gg [AH]$ is very nicely formalized in [16]. But how are these assumptions justified? In [1], Bennett suggests that such assumptions are justified by *domain specific* inference rules. In this paper we present a *domain-independent* method for justifying such assumptions. In particular, we define the *order of magnitude* of a quantity on a logarithmic scale. We show how the order of magnitude of exogenous quantities like C_a, K_w, and K_a can be propagated through a set of equations like Equations 1–4 to compute orders of magnitudes of the remaining quantities like $[H^+]$, $[OH^-]$, $[A^-]$, and $[AH]$. The computed orders of magnitude of $[A^-], [OH^-]$, and $[AH]$ can be used to justify the above assumptions and simplify the equations.

The reasoning technique described here has been implemented in a program called NAPIER.[1] NAPIER has been successfully tested on multiple examples in the domain of electromechanical devices, with the number of equations per example ranging from about 25 to a little over 150.

In the next section, we define the order of magnitude of a quantity and present rules for propagating orders of magnitudes through equations. We show that these rules can be used to infer orders of magnitudes of quantities related by a set of non-linear simultaneous equations. In Section 3 we show that, in general, order of magnitude reasoning is intractable (NP-hard). We show that the exponential blow up does occur in practice, and hence, in Section 4, we present an approximate reasoning technique that is efficient in practice. Some of the order of magnitude rules introduced in Section 2 are heuristic rules. In Section 5 we estimate the error introduced by the use of these heuristic rules, while in Section 6 we consider alternate rules that are guaranteed to be correct. We conclude with a discussion of related work.

[1] John Napier (1550–1617), a Scottish nobleman, is credited with the first discovery of logarithms.

2 Order of magnitude reasoning in NAPIER

Order of magnitude reasoning in NAPIER is a form of interval reasoning. The *order of magnitude* of a quantity q (denoted $om(q)$) is defined as follows:

$$om(q) = \lfloor \log_b |q| \rfloor \tag{5}$$

The choice of the base, b, of the logarithm is arbitrary. However, as we shall see later, it critically affects the choice of the inference rules used to propagate orders of magnitudes through equations. In this section we assume that b is the smallest number that can be considered to be "much larger" than 1. Of course, what counts as "much larger" than 1 is domain and task dependent. In our use of order of magnitude reasoning, we have assumed that 10 is the smallest number "much larger" than 1, i.e., $b = 10$. In Section 6 we consider values of b that cannot be considered to be "much larger" than 1.

Note that the order of magnitude of a quantity, q, is independent of its sign, and hence $om(q) = om(-q)$. In what follows, we assume that the signs of all quantities have been determined, to the extent possible, prior to any reasoning about orders of magnitude using standard constraint satisfaction techniques.[2]

2.1 Inference rules in NAPIER

Given the orders of magnitude of q_1 and q_2, NAPIER computes bounds on the orders of magnitude of arithmetic expressions involving q_1 and q_2, using the rules shown in Figure 3. The rules for $(q_1 + q_2)$ and $(q_1 - q_2)$ assume that q_1 and q_2 have the same sign so that the magnitudes of q_1 and q_2 are actually being added or subtracted, respectively. The rule for $(q_1 \pm q_2)$ is applicable to a sum or difference of q_1 and q_2 when the sign of at least one of q_1 and q_2 is unknown.

The rules for $(q_1 * q_2)$ and (q_1/q_2) (rules 1 and 2) follow directly from Equation 5 and the rules of interval arithmetic [10]. For example, if $om(q_1) = n_1$ and $om(q_2) = n_2$, it follows that $b^{n_1} \leq |q_1| < b^{n_1+1}$ and $b^{n_2} \leq |q_2| < b^{n_2+1}$. Using interval arithmetic, we get $b^{n_1+n_2} \leq |q_1 * q_2| < b^{n_1+n_2+2}$, and hence $n_1 + n_2 \leq om(q_1 * q_2) \leq n_1 + n_2 + 1$.

Like rules 1 and 2, rules 3a and 4a are also based on Equation 5 and interval arithmetic. Note, however, that these rules predict larger intervals for $(q_1 + q_2)$ and $(q_1 - q_2)$ than interval arithmetic predicts under the same restrictions on q_1 and q_2. For example, if $om(q_1) = om(q_2) = n$, then interval arithmetic predicts that $(q_1 + q_2)$ is bounded by $2b^n$ and $2b^{n+1}$, while NAPIER predicts $n \leq om(q_1 + q_2) \leq n + 1$, which is equivalent to saying that $(q_1 + q_2)$ lies between b^n and b^{n+2}. This larger interval is a consequence of NAPIER being able to represent only intervals whose end points are integer powers of the chosen base. Further note that rules 3a and 4a are correct only

[2]This assumption is unnecessarily strong. For example, if a and b are positive, constraint satisfaction alone is unable to deduce the sign of $a - b$. However, if $om(a) > om(b)$, then $a - b$ can be deduced to be positive.

1)	$om(q_1) + om(q_2) \leq om(q_1 * q_2) \leq om(q_1) + om(q_2) + 1$	
2)	$om(q_1) - om(q_2) - 1 \leq om(q_1/q_2) \leq om(q_1) - om(q_2)$	
3a)	$om(q_1) \leq om(q_1 + q_2) \leq om(q_1) + 1$	if $om(q_1) = om(q_2)$
b)	$om(q_1 + q_2) = om(q_1)$	if $om(q_1) > om(q_2)$
c)	$om(q_1 + q_2) = om(q_2)$	if $om(q_1) < om(q_2)$
4a)	$om(q_1 - q_2) \leq om(q_1)$	if $om(q_1) = om(q_2)$
b)	$om(q_1 - q_2) = om(q_1)$	if $om(q_1) > om(q_2)$
c)	$om(q_1 - q_2) = om(q_2)$	if $om(q_1) < om(q_2)$
5a)	$om(q_1 \pm q_2) \leq om(q_1) + 1$	if $om(q_1) = om(q_2)$
b)	$om(q_1 \pm q_2) = om(q_1)$	if $om(q_1) > om(q_2)$
c)	$om(q_1 \pm q_2) = om(q_2)$	if $om(q_1) < om(q_2)$

Figure 3: Rules for order of magnitude reasoning

if the base is greater than or equal to 2. This is reasonable given our heuristic for selecting the base (viz., 2 is unlikely to be considered to be "much larger" than 1).

Unlike the rules discussed thus far, rules 3b, 3c, 4b, and 4c are not guaranteed to be correct, but are heuristic rules. They are all based on the intuition that adding or subtracting a "small" quantity from a "large" quantity does not significantly affect the larger quantity. Since the base in Equation 5 is chosen as the smallest number that can be considered to be "much larger" than 1, the above intuition justifies these rules; the order of magnitude of a quantity is not affected by adding or subtracting quantities of a smaller order of magnitude. The inclusion of these heuristic order of magnitude rules differentiates NAPIER from standard interval reasoners. In Section 5, we estimate the error introduced by the use of these heuristic rules, while in Section 6 we consider alternate rules that do not introduce any error.

Finally, rule set 5 merely encompasses both rule sets 3 and 4. It is used to infer the order of magnitude of a sum or difference of two quantities when the signs of at least one of the two quantities is not known. To determine the order of magnitude of a sum or difference of two quantities, NAPIER selects the appropriate rule set from rule sets 3, 4, and 5, depending on the operation (sum or difference) and the signs of the two quantities. For example, consider the equation $q_3 = q_1 + q_2$. If q_1 and q_2 have the same sign, then rule set 3 is used to infer $om(q_3)$; if q_1 and q_2 have opposite signs, then rule set 4 is used to infer $om(q_3)$, since the magnitude of q_3 is really the difference of the magnitudes of q_1 and q_2; and if the signs of at least one of q_1 and q_2 is unknown, then rule set 5 is used to infer $om(q_3)$.

2.2 Set of simultaneous equations

Until now, we have focussed exclusively on how NAPIER uses a single equation to propagate orders of magnitudes, i.e., how $om(q_1 op\ q_2)$ is computed from $om(q_1)$ and $om(q_2)$. However, the rules in Figure 3 can also be used to compute orders of magnitudes of quantities related by a set of (possibly non-linear) simultaneous equations. NAPIER uses these rules to convert a set of simultaneous equations into a set of constraints, where each constraint is a disjunction of a set of linear inequalities. Each equation in the set of simultaneous equations contributes a constraint as follows:

1. Product and quotient terms contribute a single set of linear inequalities according to rules 1 and 2, respectively. For example, $q_3 = q_1 * q_2$ contributes the following set:
$$\{om(q_1) + om(q_2) \leq om(q_3),\\ om(q_3) \leq om(q_1) + om(q_2) + 1\}$$

2. Sum and difference terms contribute a disjunction of three sets of linear inequalities, using rule sets 3, 4, or 5, as applicable. Each disjunct corresponds to one of the rules (a, b, or c) in the applicable rule set. For example, assuming that q_1 and q_2 have the same sign, the equation $q_3 = q_1 - q_2$ contributes the following disjunction:[3]
$$\{om(q_3) \leq om(q_1), om(q_1) = om(q_2)\}$$
$$\vee$$
$$\{om(q_3) = om(q_1), om(q_1) \geq om(q_2) + 1\}$$
$$\vee$$
$$\{om(q_3) = om(q_2), om(q_1) \leq om(q_2) - 1\}$$
corresponding to rules 4a, 4b, and 4c, respectively.

NAPIER uses this set of constraints to compute bounds on the orders of magnitudes of the quantities. Since all the inequalities in the constraints are linear inequalities, NAPIER uses linear programming [4], in conjunction with backtracking, to compute order of magnitude bounds. Backtracking is necessary to handle the disjunctions. We describe this algorithm next.

2.3 Backtracking algorithm

Let E denote the set of simultaneous equations being processed. NAPIER's backtracking procedure is best visualized as a depth-first traversal of a backtrack tree. Each level in the tree (except the root level) corresponds to one of the sum or difference terms in E. The root level corresponds to all the product and quotient terms in E. Each internal node has three children, corresponding to the three disjuncts in

[3]Since orders of magnitudes are integral, $om(q_1) > om(q_2)$ is equivalent to $om(q_1) \geq om(q_2) + 1$.

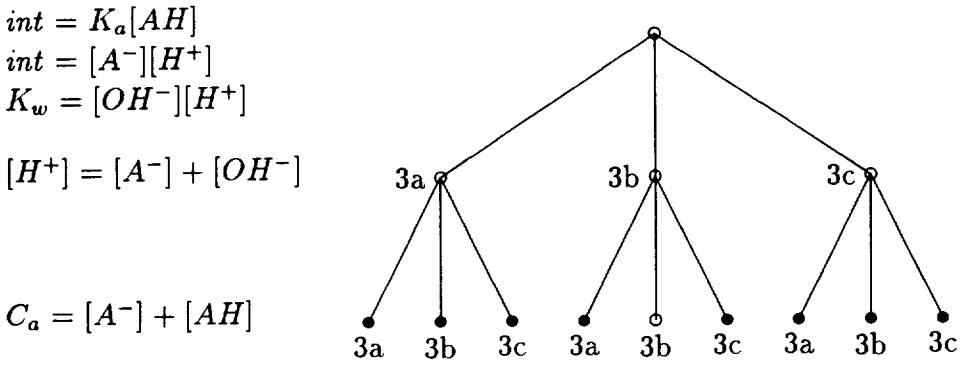

Figure 4: A backtrack tree.

the constraint contributed by the sum or difference term at the level of the node's children. Each node in the tree has an associated set of linear inequalities defined as follows:

1. The set of inequalities at the root node consists of the union of the sets of inequalities contributed by each product and quotient term in E.

2. The set of inequalities at each non-root node consists of the union of (a) the inequalities at the node's parent; and (b) the inequalities in the disjunct associated with that node.

Starting at the root node, NAPIER traverses the backtrack tree in a depth-first manner. At each node it checks the consistency of the inequalities at that node. If the set is inconsistent, it immediately backtracks to the node's parent. If the set is consistent and it is a non-leaf node, it continues its depth-first traversal. If the set is consistent and it is a leaf node, it uses the inequalities to find the maximum and minimum values of the order of magnitude of each quantity. The bounds computed at each of the consistent leaf nodes are combined so that the lower bound of each quantity is the least lower bound and the upper bound is the greatest upper bound.

Since the inequalities at each node are linear, NAPIER uses the Simplex linear programming algorithm [4; 14] to check their consistency, and to compute the order of magnitude bounds at leaf nodes. However, from Equation 5 it follows that the order of magnitude of a quantity is integral. Hence, instead of using linear programming, NAPIER should use integer programming [4]. Unfortunately, integer programming is known to be intractable [6], which leads to severe restrictions on the number of equations and the size of the backtrack tree that can be handled. Hence, to avoid such restrictions, NAPIER uses linear programming.

It is important to note that, while bounds computed by linear programming are not guaranteed to be tight[4], they are guaranteed to be correct: upper bounds will be

[4]A bound b_1, interpreted as an interval, is said to be *tight* with respect to a bound b_2 if b_1 and b_2 are identical. b_1 is *looser* than b_2 if b_1 contains b_2.

greater than or equal to integer programming upper bounds, and lower bounds will be less than or equal integer programming lower bounds. In addition, we have found that, in practice, linear programming bounds are usually integral, in which case there is no loss of solution quality.

2.4 Example

We now illustrate the above procedure using Equations 1–4. Let us assume that $b = 10$ and the exogenous orders of magnitude are as follows: $om(K_w) = -14, om(K_a) = -2, om(C_a) = -5$. This corresponds to a moderately strong solution of a strong acid. The backtrack tree resulting from these equations is shown in Figure 4. The equations associated with each level are shown on the left of the tree. Note that Equation 3 had to be split into two product expressions, with the introduction of an intermediate variable *int*. The rules (and hence the disjuncts) associated with each non-root node are displayed near each node. Nodes that are filled in are the inconsistent nodes. For example, the left most leaf node can be seen to be inconsistent using the following line of reasoning. Applying rule 3a to Equations 1 and 2, we get

$$om([OH^-]) = om([A^-]) = om([AH])$$
$$om([A^-]) \leq om(C_a) \leq om([A^-]) + 1$$
$$om([A^-]) \leq om([H^+]) \leq om([A^-]) + 1$$

Since $om(C_a) = -5$, it follows that the least value of $om([A^-])$ is -6. Hence the least values of $om([OH^-])$ and $om([H^+])$ are also -6, and hence the least value of $om([OH^-][H^+])$ is -12. But rule 1 applied to Equation 4 requires that:

$$om([OH^-][H^+]) = om(K_w) = -14$$

which leads to a contradiction.

Of course, NAPIER doesn't need the above line of reasoning to infer inconsistencies; it reaches the same conclusion using linear programming.

The only consistent set of inequalities at the leaf nodes is the middle most leaf node, corresponding to assuming that $om([A^-]) > om([OH^-])$ and $om([A^-]) > om([AH])$. The quantity bounds calculated at this node are as follows:[5]

$$\begin{aligned} om([H^+]) &= -5 \\ om([OH^-]) &= (-10, -9) \\ om([AH]) &= (-9, -7) \\ om([A^-]) &= -5 \end{aligned}$$

Since $[A^-]$ is at least two orders of magnitude greater than $[AH]$, and at least four orders of magnitude greater than $[OH^-]$, a chemist is justified in making the assumptions that $[A^-] \gg [OH^-]$ and $[A^-] \gg [AH]$. These assumptions can then be used to simplify the equations, as discussed earlier.

[5] $om(q) = (l, u)$ represents the fact that $l \leq om(q) \leq u$

A slight variation of the above example illustrates the importance of having such justifications. Suppose that, instead of having $om(C_a) = -5$, we had $om(C_a) = -8$. This corresponds to a weak solution of the same strong acid. Using this new value for $om(C_a)$, NAPIER predicts the following bounds on the orders of magnitude:

$$\begin{aligned} om([H^+]) &= -7 \\ om([OH^-]) &= (-8, -7) \\ om([AH]) &= (-14, -12) \\ om([A^-]) &= -8 \end{aligned}$$

These values justify the assumption that $[A^-] \gg [AH]$, but the other assumption, $[A^-] \gg [OH^-]$, is seen to be completely unjustified. This means that only Equation 2 can be simplified. Hence, NAPIER is a useful tool in justifying the order of magnitude assumptions that scientists and engineers make in simplifying equations.

In addition to its role in justifying order of magnitude assumptions, NAPIER's predictions can also be used directly. For example, if all the chemist is interested in is the approximate pH of the solution[6], then NAPIER's predictions can be used directly: in the first case, the pH is between 5 and 4; in the second case, the pH is between 7 and 6. Note that NAPIER was able to make these predictions using approximate values of C_a, K_w, and K_a. This feature makes it particularly useful during conceptual design.

3 Order of magnitude reasoning is intractable

The backtracking algorithm described in the previous section, generates a tree whose worst case size is exponential in the number of sum and difference expressions. In this section we show that order of magnitude reasoning using the rules in Figure 3 is intractable, even if orders of magnitude are not required to be integral. This means that NAPIER can do little better than generate a backtrack tree whose worst case size is exponential.

We start by defining the decision problem corresponding to finding the maximum order of magnitude of a quantity:

Definition 1 *(*ORDER OF MAGNITUDE REASONING*) Let E be a set of equations, and let V be the set of quantities used in E. Let $X \subseteq V$ be the set of exogenous quantities, with known orders of magnitude. Let $q \in V$ be a quantity and let B be an integer. Let $s : V \rightarrow \{+, -, unknown\}$ be a function that assigns signs to the quantities in V. (Quantities with unknown signs are assigned "unknown.") Assuming that the order of magnitude of a quantity is not required to be integral, is the maximum value of $om(q)$, derived using the rules in Figure 3 on the set E, greater than or equal to B?*

[6]The pH of a solution is defined to be $-\log_{10}[H^+]$.

The following theorem states the intractability of order of magnitude reasoning.[7]

Theorem 1 *The* ORDER OF MAGNITUDE REASONING *problem is NP-complete.*

Proof: See Appendix A. □

Assuming $P \neq NP$, Theorem 1 tells us that, in the worst case, NAPIER will have to generate a backtrack tree whose size is exponential in the number of sum and difference terms. Unfortunately, the exponential blow up does occur in practice. We have used NAPIER in the domain of electromechanical devices as part of the automated model selection system described in [12; 13]. Table 1 summarizes NAPIER's performance on models of ten different devices (see [12] for a description of the devices).

Example number	Number of equations	Number of +/− terms	Time (sec) on an Explorer II	
			All equations	With causal ordering
1	28	11	2733	2.0
2	31	11	2435	1.0
3	45	14	–	2.9
4	60	24	–	2.7
5	80	25	–	37.2
6	110	32	–	35.9
7	111	32	–	94.6
8	119	35	–	20.4
9	145	43	–	45.2
10	163	50	–	21.0

Table 1: NAPIER's run times with and without causal ordering.

The second column in this table shows the total number of equations in each example, while the third column shows the the total number of sum and difference terms. The fourth column shows the time it took NAPIER to run its backtracking algorithm on the complete set of equations.[8] NAPIER was given a maximum of one hour to solve each example; a "–" entry in column four denotes that NAPIER could not solve the example in an hour. As is clear from the table, only the two smallest examples could be solved in under an hour, each taking over 40 minutes. Hence, NAPIER appears to be quite impractical, except for the smallest examples. To make it practical, we now develop an approximate reasoning scheme for NAPIER that trades off accuracy for speed.

[7] See [3] for a comprehensive introduction to the theory of intractability.

[8] The fifth column will be discussed in the next section.

4 Approximation algorithms in NAPIER

The backtrack tree developed by NAPIER is, in the worst case, exponential in the number of sum and difference terms in the set of equations under consideration. Hence, to make NAPIER practically useful, it is important to decrease the number of sum and difference terms that are handled at any one time. We now discuss a method for doing this, based on a *dependency ordering* of the equations.

4.1 Ordering the equations

The dependency ordering of equations that we consider is the *causal ordering*, described in [5]. The causal ordering specifies the order in which equations are to be solved, and identifies minimal sets of equations that *must* be solved simultaneously. The causal ordering can be viewed as a directed acyclic graph. Each node in the graph consists of a set of equations that must be solved simultaneously. There is an edge from node n_1 to node n_2 if the equations at n_2 use a quantity whose value is determined by the equations at n_1.

NAPIER processes the equation sets in the order specified by the causal ordering: equation sets earlier in the ordering are processed first. NAPIER bounds the orders of magnitudes of the quantities used in an equation set, and uses these bounds as exogenous bounds for equation sets later in the ordering.

The use of the above dependency ordering has a significant computational advantage. A large set of equations, with many sum and difference terms, can often be broken down into many small sets of equations, with each equation set having very few sum and difference terms. Hence, NAPIER can process each equation set in the dependency ordering very fast. Column five in Table 1 shows the time it took NAPIER to solve the ten examples using causal ordering. It takes NAPIER from a few seconds to under two minutes to solve each of these examples, showing that causal ordering has made NAPIER practical for large sets of equations.

4.2 Loss of accuracy

The drawback of using the dependency ordering is that global constraints can be lost, leading to excessively loose bounds on the orders of magnitudes. Consider, for example, the set $\{y_1 = x_1 * x_2, y_2 = x_3/y_1, y_3 = y_1 * y_2\}$, and let x_1, x_2, and x_3 be exogenous with orders of magnitude 0. The dependency ordering generated from this set of equations is:

$$\begin{array}{ccc} & \{y_2 = x_3/y_1\} & \\ \nearrow & & \searrow \\ \{y_1 = x_1 * x_2\} & \longrightarrow & \{y_3 = y_1 * y_2\} \end{array}$$

Using this dependency ordering, NAPIER computes the order of magnitude of y_3 as follows: from the first equation it computes $om(y_1)$ to be between 0 and 1; from the second equation, and the calculated bound on $om(y_1)$, it computes $om(y_2)$ to be

Example #	Δ_{max}	Example #	Δ_{max}
1	11	6	12
2	11	7	7
3	10	8	9
4	9	9	7
5	9	10	9

Table 2: Maximum value of Δ for each example.

between -2 and 0; and from the third equation and the calculated bounds on $om(y_1)$ and $om(y_2)$, it computes $om(y_3)$ to be between -2 and 2. However, if all three equations were considered simultaneously, NAPIER computes $om(y_3)$ to be between -1 and 1.

The reason for the looser bound in the first case stems from not enforcing some global constraints. For example, the lower bound of $om(y_3)$ can be -2 only when $om(y_1) = 0$ and $om(y_2) = -2$. However, when $om(y_1)$ is 0, the second equation dictates that the lowest that $om(y_2)$ can be is -1. This fact is lost when the third equation is processed by itself.

More generally, the above problem occurs when a quantity, like y_3, depends on two or more quantities, like y_1 and y_2, whose values have been determined by equations that are earlier in the causal ordering. In using these previously determined values, NAPIER disregards any additional constraints that might hold between those values. Hence, bounds computed based on these values may not be as tight as possible.

NAPIER can partially address this problem by combining adjacent sets of equations in the dependency ordering. This allows more equations to be handled simultaneously, so that more global constraints can be incorporated. However, combining adjacent sets of equations can lead to an increase in the number of sum and difference terms that must be handled simultaneously. Hence, adjacent sets are combined only when the number of sum and difference terms in the resulting set does not increase beyond a threshold (call this threshold Δ).

Combining adjacent sets of equations, as described above, also allows us to partially empirically evaluate the effect of causal ordering on accuracy. We ran NAPIER a number of times on each of our examples, using increasing values of Δ, allowing a maximum of one hour per run. Table 2 shows the maximum value of Δ used for each example. We then compared the bounds that were computed without combining adjacent sets with the bounds that were computed with the maximum setting of Δ. Interestingly, we found that there was *no* loss of accuracy—the bounds computed with and without combining adjacent sets were identical.

To understand the reason for this somewhat surprising result, we now analyze the source of the additional constraints on previously determined values. Let us assume that $om(p_3)$ is computed using previously computed values of $om(p_1)$ and $om(p_2)$. Additional constraints on the values of $om(p_1)$ and $om(p_2)$ stem from one of two sources: (a) $om(p_1)$ and $om(p_2)$ are determined simultaneously; and (b) the value

Example number	Equations per node		# of extra edges
	Maximum	Average	
1	7	1.27	1
2	7	1.24	0
3	7	1.15	1
4	1	1.00	0
5	12	1.29	1
6	18	1.29	2
7	17	1.26	6
8	9	1.25	2
9	18	1.21	3
10	16	1.10	0

Table 3: Properties of the causal ordering graph

of $om(p_1)$ is used in computing the value of $om(p_2)$, i.e., the values of one of these quantities depends on the value of the other. Point (a) manifests itself as a node in the causal ordering which contains more than one equation. Point (b) manifests itself as multiple paths between two nodes in the causal ordering.

Hence, if the causal ordering, viewed as a graph, satisfies the following two properties:

1. each node contains exactly one equation; and

2. there is at most one path between any two nodes;

then we can show that there will be *no* additional constraints between previously determined values. Hence, there is no loss of accuracy in using the causal ordering.

Table 3 shows how closely the causal orderings generated from our examples match the above two properties. The second and third columns of this table show the maximum and average number of equations per node, respectively. One can see that, in all cases, the average number of equations per node is very close to 1. The fourth column shows the minimum number of edges that must be removed from the causal ordering to ensure that there is at most one path between any two nodes. One can see that, in most cases these numbers are very small. Hence, the above analysis provides us with some insight into the reasons underlying the fact that, in our examples, the bounds computed with and without combining adjacent sets are identical.

5 Error estimation

In this section, we estimate the error introduced by the use of the heuristic rules introduced in Section 2.1. We then analyze some alternate order of magnitude rules that seem intuitively plausible, and show that these rules introduce unacceptably large

errors. The analysis is done using probability theory and is based on interpreting each quantity as a *random variable*.[9]

5.1 Estimating the error of heuristic rules

We start by analyzing rule 3b. Let Q, Q_1, and Q_2 be quantities such that $Q = Q_1 + Q_2$. Let f_{Q_1} and f_{Q_2} be the probability density functions of Q_1 and Q_2, respectively, and let f_{Q_1,Q_2} be their joint probability density function. (Briefly, $f_{Q_1}(q_1)$ is the probability that Q_1 lies between q_1 and $q_1 + dq_1$, and $f_{Q_1,Q_2}(q_1, q_2)$ is the probability that Q_1 lies between q_1 and $q_1 + dq_1$, and Q_2 lies between q_2 and $q_2 + dq_2$.) Since $Q = Q_1 + Q_2$, it follows that the probability that Q lies between l and u, for any values l and u, is:

$$Prob\{l \leq Q < u\} = \int_{-\infty}^{\infty} \int_{l-q_1}^{u-q_1} f_{Q_1,Q_2}(q_1, q_2) dq_2 dq_1 \tag{6}$$

Let us now assume that $om(Q_1) = n_1$ and $om(Q_2) = n_2$, with $n_1 > n_2$. Under these conditions, rule 3b states that $om(Q) = n_1$, i.e., $b^{n_1} \leq Q < b^{n_1+1}$. To estimate the error in rule 3b, ϵ(Rule 3b), we must calculate the probability that Q lies outside the region from b^{n_1} to b^{n_1+1}:

$$\begin{aligned} &\epsilon(\text{Rule 3b}) \\ &= 1 - Prob\{b^{n_1} \leq Q < b^{n_1+1}\} \\ &= 1 - \int_{-\infty}^{\infty} \int_{b^{n_1}-q_1}^{b^{n_1+1}-q_1} f_{Q_1,Q_2}(q_1, q_2) dq_2 dq_1 \end{aligned} \tag{7}$$

To evaluate this integral, we make the following assumptions:

Assumption 1: Q_1 and Q_2 are independent random variables. Hence, the joint probability density of Q_1 and Q_2 is just the product of the individual probability densities:

$$f_{Q_1,Q_2}(q_1, q_2) = f_{Q_1}(q_1) f_{Q_2}(q_2) \tag{8}$$

Assumption 2: Q_1 and Q_2 are uniformly distributed on the intervals $[b^{n_1}, b^{n_1+1})$ and $[b^{n_2}, b^{n_2+1})$, respectively:

$$f_{Q_1}(q_1) = \begin{cases} \frac{1}{b^{n_1+1} - b^{n_1}} & \text{if } b^{n_1} \leq q_1 < b^{n_1+1} \\ 0 & \text{otherwise} \end{cases}$$

$$f_{Q_2}(q_2) = \begin{cases} \frac{1}{b^{n_2+1} - b^{n_2}} & \text{if } b^{n_2} \leq q_2 < b^{n_2+1} \\ 0 & \text{otherwise} \end{cases}$$

[9] See [2] for an introduction to probability theory and random variables.

$$
\begin{array}{lrcl}
1') & om(q_1 * q_2) & = & om(q_1) + om(q_2) \\
2') & om(q_1/q_2) & = & om(q_1) - om(q_2) \\
3a') & om(q_1 + q_2) & = & om(q_1) \text{ if } om(q_1) = om(q_2)
\end{array}
$$

Figure 5: Alternate rules for order of magnitude reasoning.

Using these assumptions, we get the following result (see Appendix B for an outline of the derivation):

$$\epsilon(\text{Rule 3b}) = \frac{b+1}{2b^{n_1-n_2}(b-1)} \tag{9}$$

Hence, under Assumptions 1 and 2, the error in rule 3b is maximum when $(n_1 - n_2)$ is minimum, i.e., $(n_1 - n_2) = 1$, which occurs when quantities of consecutive orders of magnitude are being added. When $b = 10$, the maximum error is 6.11%. For larger values of either $(n_1 - n_2)$ or b, the errors are even smaller. For example, with $b = 10$, and $(n_1 - n_2) = 2$ (i.e., adding a quantity that is two orders of magnitude smaller) the estimated error is only 0.61%. The error in rule 4b can also be shown to be $(b+1)/2b^{n_1-n_2}(b-1)$ in a similar way.

5.2 Alternate order of magnitude rules

The above error estimation techniques can also be used to analyze alternate rules for order of magnitude reasoning. In particular, we analyze the three inference rules shown in Figure 5. These rules were our first attempt at modeling an engineers order of magnitude reasoning. Rule 1′ and 2′ were meant to model reasoning like: "If the resistance, R, is about 10^{-1} ohms, and the current, i, is about 10^{-2} amps, then the voltage drop, V $(= iR)$, is about 10^{-3} volts." The idea was that the order of magnitude of a product or quotient was the sum or difference, respectively, of the orders of magnitudes of the arguments. Rule 3a′ was meant to model the intuition that adding quantities of the same order of magnitude results in a quantity of the same order of magnitude. However, we can show that, while these rules may appear intuitively appealing, they are also unacceptably error-prone. In particular, we can use the above error estimation techniques to show that:

$$\epsilon(\text{Rule 1'}) = 1 - (b\ln b - b + 1)/(b-1)^2 \tag{10}$$

$$\epsilon(\text{Rule 2'}) = 1/2 \tag{11}$$

$$\epsilon(\text{Rule 3a'}) = 1 - (b-2)^2/2(b-1)^2 \tag{12}$$

Substituting $b = 10$ into the above equations tells us that the error in rule 1′ is 82.68%, the error in rule 2′ is 50%, and the error in rule 3a′ is 60.49%. We believe that these errors are unacceptably large, and hence have chosen not to include these rules in NAPIER.

5.3 Discussion

The error estimation results presented above depend crucially on Assumptions 1 and 2. Assumption 1 assumes that the two quantities being combined, Q_1 and Q_2, are independent random variables. This assumption is reasonable if Q_1 and Q_2 are exogenous quantities. It is also reasonable if the set of exogenous quantities used to calculate the order of magnitude of Q_1 is disjoint from the set of exogenous quantities used to calculate the order of magnitude of Q_2. However, if the orders of magnitudes of Q_1 and Q_2 depend on the order of magnitude of a common exogenous quantities, then Q_1 and Q_2 are not independent.

Assumption 2 assumes that the two quantities being combined, Q_1 and Q_2, are uniformly distributed random variables. In the absence of any additional information, this assumption is reasonable for exogenous quantities. However, it breaks down for derived quantities. For example, if Q_1 and Q_2 are uniformly distributed random variables, and if $Q = Q_1\ op\ Q_2$ (where op is one of $+, -, *, \text{or} /$), then Q is *not* uniformly distributed. Hence, when the order of magnitude of Q is used to calculate the orders of magnitudes of other quantities, Assumption 2 is not valid.

The above discussion implies that our error estimation technique has limited applicability. In particular, the errors estimated in this section cannot be directly used to estimate the error introduced in predictions based on a set of equations. Nonetheless, these techniques have proved useful in helping us select a reasonable set of order of magnitude reasoning rules, while alerting us to the possibility of large errors introduced by alternate rules.

6 Order of magnitude reasoning without heuristic rules

The error analysis in the previous section was necessary because of our use of heuristic order of magnitude rules (rules 3b, 3c, 4b, 4c, 5b, and 5c). The error introduced by these rules can be removed by replacing these rules with the rules shown in Figure 6 (replace rule 3b with rule 3b′, etc.). Using interval arithmetic and Equation 5, one can see that these rules introduce no error, and are guaranteed to be correct. However, their drawback is that, like rules 3a, 4a, and 5a, they predict larger intervals for $(q_1 + q_2)$, $(q_1 - q_2)$, and $(q_1 \pm q_2)$ than interval arithmetic predicts under the same restrictions on q_1 and q_2. For example, if $om(q_1) = n$, and $om(q_2) = n - 1$, then interval arithmetic predicts that $(q_1 + q_2)$ is bounded by $b^n + b^{n-1}$ and $b^{n+1} + b^n$, while rule 3b′ predicts that $(q_1 + q_2)$ lies in the larger interval from b^n and b^{n+2}.

Let R_1 be the set of rules in Figure 3, and let R_2 be the result of replacing rules 3b, 3c, 4b, 4c, 5b, 5c in R_1 by the rules in Figure 6. Clearly, NAPIER with R_2 predicts larger intervals than with R_1. To empirically compare the interval widths predicted by R_1 and R_2 when $b = 10$, we ran NAPIER with the two sets of rules on the ten examples introduced earlier (Table 1). For each quantity, q, in each example, we compared the

$$
\begin{array}{lll}
3b') & om(q_1) \leq om(q_1 + q_2) \leq om(q_1) + 1 & \text{if } om(q_1) > om(q_2) \\
3c') & om(q_2) \leq om(q_1 + q_2) \leq om(q_2) + 1 & \text{if } om(q_1) < om(q_2) \\
 & & \\
4b') & om(q_1) - 1 \leq om(q_1 - q_2) \leq om(q_1) & \text{if } om(q_1) > om(q_2) \\
4c') & om(q_2) - 1 \leq om(q_1 - q_2) \leq om(q_2) & \text{if } om(q_1) < om(q_2) \\
 & & \\
5b') & om(q_1) - 1 \leq om(q_1 - q_2) \leq om(q_1) + 1 & \text{if } om(q_1) > om(q_2) \\
5c') & om(q_2) - 1 \leq om(q_1 - q_2) \leq om(q_2) + 1 & \text{if } om(q_1) < om(q_2)
\end{array}
$$

Figure 6: Correct rules for order of magnitude reasoning.

Example Number	$R_2^{10}(q)/R_1^{10}(q)$	$R_3^2(q)/R_1^{10}(q)$ Exo. ratio 0.98	$R_3^2(q)/R_1^{10}(q)$ Exo. ratio 0.16	$R_3^2(q)/R_1^{10}(q)$ Exo. ratio 0.62
1	3.90	1.28	0.11	0.54
2	1.59	1.07	0.09	0.41
3	48.81	1.37	0.12	0.62
4	5.2e6	2.37	0.15	1.12
5	2.6e4	2.58	0.51	1.44
6	1.9e3	1.10	0.10	0.43
7	29.8	1.00	0.35	0.62
8	1.8e6	1.79	0.14	0.81
9	1.5e9	9.00	0.37	4.12
10	1.5e8	17.43	0.41	5.15

Table 4: This table compares different rule sets and bases. $R_1^{10}(q)$ is the predicted interval width of q with rule set R_1 and base 10. Similarly, $R_2^{10}(q)$ uses rule set R_2 and base 10, and $R_3^2(q)$ uses rule set R_3 and base 2. Columns two through five show the average values of the ratios displayed at the top of each column for each example. The exogenous ratios specified at the top of the last three columns are the average ratios of the width of an exogenous parameter with base 2 to the width with base 10.

width, $R_2^{10}(q)$, of the interval predicted by R_2 using base 10, with the width, $R_1^{10}(q)$, of the interval predicated by R_1 using base 10. To ensure a fair comparison, the orders of magnitudes of exogenous quantities were the same for both sets of rules. The second column in Table 4 summarizes our results. This column displays the average of the ratios $R_2^{10}(q)/R_1^{10}(q)$ in each example, where q ranges over the set of non-exogenous quantities in the example. Not surprisingly, R_2 always predicts larger intervals. Unfortunately, these intervals are often significantly larger (up to 10^9 times as large), making rule set R_2 with base 10 largely useless in practice. The heuristic rules in R_1 are useful precisely because they prevent this interval explosion, without introducing too much error.

$$3a'') \quad om(q_1 + q_2) = om(q_1) + 1 \quad \text{if } om(q_1) = om(q_2)$$

$$4a'') \quad om(q_1 - q_2) \leq om(q_1) - 1 \quad \text{if } om(q_1) = om(q_2)$$

Figure 7: Tighter rules for base 2.

While $R_2^{10}(q)$ will always be larger than $R_1^{10}(q)$, the former is, on the average, significantly larger than the latter in the above experiment because the base of the logarithm is quite large ($b = 10$). In particular, one application of rule 3b′ makes the interval width of a sum be approximately b times larger than either applying rule 3b or using interval arithmetic. Hence, larger the value of b, larger the predicted interval widths. This suggests using smaller values of b. Note that we are free to use smaller values of b because we are not using heuristic rules, so that we do not require b to be "much larger" than 1.

A natural choice for a smaller b is 2, because it is the smallest number that ensures that rule 3a continues to be correct (see Section 2.1).[10] In fact, using $b = 2$, we can further strengthen rules 3a and 4a with rules 3a″ and 4a″ shown in Figure 7. One can verify that, with $b = 2$, the predictions of rules 3a″ and 4a″ are identical to the predictions of interval arithmetic; predictions of rules 3a and 4a are looser.

Let R_3 be the set of rules resulting from replacing rules 3a and 4a with rules 3a″ and 4a″ in R_2. To empirically compare the interval widths predicted by R_1 (using base 10) and R_3 (using base 2), we ran NAPIER with the two sets of rules on our ten examples. As before, for each quantity, q, in each example, we compared the width, $R_3^2(q)$, of the interval predicted by R_3 using base 2, with the width, $R_1^{10}(q)$, of the interval predicted by R_1 using base 10. To ensure a fair comparison, we made sure that exogenous values in both cases were approximately equal; the exogenous values cannot be exactly equal because the bases are different in the two cases. For example, if the exogenous value of $om(q)$ using base 10 is 1 (i.e., $10 \leq q < 100$), then with base 2 we would use $4 \leq om(q) \leq 6$ (i.e., $16 \leq q < 128$). The best approximation for each exogenous value led to the width of the exogenous values with base 2 being on the average 0.98 times the width with base 10. The third column displays the average of the ratios $R_3^2(q)/R_1^{10}(q)$ in each example, where q ranges over the set of non-exogenous quantities in the example. Once again, the intervals predicted with R_3 are larger than those predicted by R_1. However, unlike the results with R_2, the intervals do not explode—the intervals range from 1 to about 17 times larger, with an average of about 2.4 times larger. This makes R_3 with base 2 useful for determining conservative upper bounds for interval widths.

A second important advantage of using base 2 is that it leads to finer granularity on the logarithmic scale than base 10. Since NAPIER is unable to exactly represent

[10] Values of b smaller than 2 would require more choices for sum and difference terms, making the backtrack tree larger and more expensive to traverse.

intervals whose end points are not integer powers of the chosen base, it must approximate them. The smaller the chosen base, the better is the approximation. For example, an exogenous quantity, q, might be known to lie between 20 and 25. If the chosen base is 10, then this must be approximated as $om(q) = 1$ (i.e., $10 \leq q < 100$). On the other hand, if the chosen base is 2, then it can be approximated more accurately as $om(q) = 4$ (i.e., $16 \leq q < 32$). Better approximations of exogenous quantities can lead to tighter bounds for non-exogenous quantities.

To empirically support this claim, we repeated the previous experiment, except that we used narrower exogenous intervals for base 2 than for base 10. The fourth column shows average $R_3^2(q)/R_1^{10}(q)$ ratios when exogenous values with base 2 are on the average 0.16 times the exogenous values with base 10. One can see that the significantly narrower exogenous intervals lead to significantly narrower predicted intervals. The fifth column shows average $R_3^2(q)/R_1^{10}(q)$ ratios when exogenous values with base 2 are on the average 0.62 times the exogenous values with base 10. Here the results are mixed: in some cases base 2 gives tighter bounds, while in other cases base 10 is tighter. This demonstrates that as the exogenous value ratios increase, the heuristic rules in R_1 become more useful.

The results of this section are helpful in selecting appropriate rule sets and bases for order of magnitude reasoning. The rule set to be used depends on the selected base as follows. If a large base, such as 10, is used then rule set R_1 should be used. Rule set R_2 is useless with large bases because of interval explosion. With a large base, the error introduced by the heuristic rules in R_1 is small, and it decreases as the base becomes larger. If a small base is used, then rule set R_2 is appropriate since the interval explosion is not so severe. Furthermore, rule set R_1 is inappropriate because the error introduced by the heuristic rules is large for small bases. When base 2 is used, rule set R_3 is preferred over R_2 because the tighter rules in Figure 7 can be used.

The choice of the base is dependent upon the characteristics of the problem to be solved. If a problem requires conservative and error-free estimates, then rule set R_1 is inapplicable, and a small base such as 2 should be selected. However, if the error introduced by using the heuristic rules in R_1 is acceptable, then the choice of base hinges on how well the bases can approximate the exogenous values of the problem. If the exogenous values are such that a small base can approximate them significantly better than a large base, then the small base is to be preferred since it will predict tighter bounds. If, on the other hand, the approximations with both the small and the large base are comparable, then the large base is preferable since the heuristic rules in R_1 will help predict tighter bounds.

7 Related work

Order of magnitude reasoning has been widely studied in AI. Murthy [11] was the first to propose the use of a logarithmic scale for the order of magnitude of a quantity.

In that paper, he also provides rules of inference to infer new orders of magnitude from old ones. Some of these rules are similar to ours. For example, he includes rules 3b, 3c, 4b, and 4c. However, instead of 1, he proposes the rule $om(q_1 * q_2) = om(q_1) + om(q_2)$ (which is rule 1′), and instead of rule 3a, he proposes the rule $om(q_1 + q_2) = om(q_1)$ when $om(q_1) = om(q_2)$ (which is rule 3a′). As we saw in Section 5.2, the estimated error in these rules is too large, and hence we have chosen not to include them in NAPIER. Unlike our work, Murthy provides no analysis of how his inference rules can be used to find the order of magnitudes of quantities related by sets of simultaneous equations. In addition, we also analyze the complexity of order of magnitude inference, and present an approximate reasoning technique that works well in practice.

Raiman [15; 16] explores the foundations of symbolic order of magnitude reasoning. He defines a variety of order of magnitude scales, such as *Close* and *Comparable*, built out of the basic order of magnitude granularities, *Small* and *Rough*. He introduces ESTIMATES, a system to solve order of magnitude equations. The primary difference between NAPIER and ESTIMATES is one of emphasis: NAPIER can be viewed as providing justifications for making order of magnitude assumptions; ESTIMATES can be viewed as a formalization of the use of such order of magnitude assumptions to symbolically manipulate and simplify equations.

Order of magnitude reasoning in the O(M) formalism [9] uses a quantity e to represent the largest quantity that can be considered to be "much smaller" than 1. This is analogous to the quantity b in NAPIER (i.e., $b = 1/e$). However, there are a number of differences between O(M) and NAPIER. First, the O(M) formalism is based on order of magnitude relations *between* quantities. Hence, it works best when equations involve only *links* (links are ratios of quantities). NAPIER, on the other hand, is based on the order of magnitudes of the quantities themselves, and hence works with any algebraic equations. This is advantageous because it is not always possible to convert equations into equations involving only links. Second, O(M) requires equations to be converted into *assignments*, which allow a new relation or range to be inferred from already known relations. This is a serious restriction since equations can be converted to assignments only in the absence of simultaneous equations. As we have seen, NAPIER does not have this restriction.

NAPIER is also related to interval reasoning discussed in [10; 17; 18]. NAPIER can be viewed as interval reasoning in which the end points of the interval are restricted to a particular set of points of the form b^n, with specified base b, and any integer n. The drawback of this restriction is that under certain conditions, compared to interval reasoning, the bounds inferred by NAPIER are unnecessarily loose (e.g., see the discussion of rule 3a in Section 2.1). The advantage of this restriction is that, unlike traditional interval reasoners, NAPIER is able to use sets of non-linear simultaneous equations to infer quantity bounds. In addition, the ability to simultaneously process all the equations in a set allows NAPIER to exploit global constraints to compute tighter bounds (see Section 4). Another distinguishing characteristic of NAPIER, which classifies it as an order of magnitude reasoning system rather than just an

interval reasoner, is its ability to use heuristic rules (e.g, rule 3b).

8 Conclusions

In this paper we described an implemented order of magnitude reasoning system called NAPIER. NAPIER defines the order of magnitude of a quantity on a logarithmic scale and uses a set of rules to propagate order of magnitudes through equations. A novel feature of NAPIER is its handling of non-linear simultaneous equations. Since the order of magnitude reasoning rules are all disjunctions of linear inequalities, NAPIER is able to use linear programming, in conjunction with backtracking, to find bounds on the order of magnitudes of quantities related by sets of non-linear simultaneous equations.

We also showed that order of magnitude reasoning using NAPIER's rules is intractable. Hence, NAPIER uses an approximate reasoning technique, based on causal ordering, leading to a practically useful system. This approximate reasoning technique trades off accuracy for speed. However, in practice, there does not appear to be any loss of accuracy.

Since some of NAPIER's rules are heuristic rules, and we estimated the error introduced by the use of these rules by interpreting each quantity as a random variable. We used the same error estimation technique to show that other intuitively appealing heuristic rules can lead to larger errors. Finally, we empirically analyzed correct alternatives to the heuristic rules. While these alternative rules do not introduce any error, they can lead to interval explosions. Interval explosion is addressed by using smaller values of the base of the logarithmic scale. The empirical analysis leads to a set of guidelines on how to select the base of the logarithm and the set of inference rules.

The work described in this paper can be extended in a number of ways. First, we would like to extend NAPIER's inference rules to handle non-algebraic operators, such as trigonometric functions. A straightforward method of doing this is to use Taylor series expansions of the operators, though care must be taken to ensure that the series expansion is truncated at the right point. Second, we would like to use NAPIER in temporal simulation. One alternative is to follow Weld's work on HR-QSIM [19], while another alternative is to follow Kuipers and Berleant's work on Q2 [8].

NAPIER has been extensively used in an automated model selection system described in [12; 13]. We believe that NAPIER will find wide spread applications in different aspects of engineering and scientific problem solving.

Acknowledgements

I would like to thank Richard Fikes, Pat Hayes, Leo Joskowicz, and Dan Weld for useful discussions and comments on earlier drafts of this paper. Thanks also to the anonymous reviewers of conference versions of this paper, whose detailed comments

helped improve the paper. Pandurang Nayak was supported by an IBM Graduate Technical Fellowship. Additional support for this research was provided by the Defense Advanced Research Projects Agency under NASA Grant NAG 2-581 (under ARPA order number 6822), by NASA under NASA Grant NCC 2-537, and by IBM under agreement number 14780042.

A Proof of Theorem 1

It is easy to see that the ORDER OF MAGNITUDE REASONING problem is in NP since a non-deterministic algorithm can proceed by (a) for each sum and difference term in E, guessing a rule (a, b, or c) from rule sets 3, 4, or 5, as applicable; and (b) use linear programming on the resulting set of inequalities to see if the maximum value of $om(q)$ exceeds B. Since linear programming is known to be in P [7], it follows that the ORDER OF MAGNITUDE REASONING problem is in NP.

To show that the ORDER OF MAGNITUDE REASONING problem is NP-hard, we reduce an arbitrary instance of 3SAT to an instance of the ORDER OF MAGNITUDE REASONING problem. Let $\mathcal{I}_1$ be an arbitrary instance of 3SAT consisting of a set $U = \{u_1, \ldots, u_n\}$ of boolean variables, and a set $C = \{c_1, \ldots, c_m\}$ of three literal clauses. We construct an instance, $\mathcal{I}_2$, of the ORDER OF MAGNITUDE REASONING problem as follows.

For each boolean variable $u_i \in U$, $1 \leq i \leq n$, add the following 6 equations to E, and the corresponding quantities to V:

$$v_i = x_{i1} * x_{i2} \tag{13}$$

$$\bar{v_i} = \bar{x_{i1}} * \bar{x_{i2}} \tag{14}$$

$$y_{i1} = v_i * \bar{v_i} \tag{15}$$

$$y_i = y_{i1} + y_{i2} \tag{16}$$

$$z_i = v_i - \bar{v_i} \tag{17}$$

$$z_i = (z_{i1} + z_{i2}) * z_{i3} \tag{18}$$

Add $x_{i1}, x_{i2}, \bar{x_{i1}}, \bar{x_{i2}}, y_i, z_{i1}$, and z_{i3} to the set X of exogenous quantities. Define the orders of magnitudes of these quantities as follows:

$$om(x_{i1}) = om(x_{i2}) = om(\bar{x_{i1}}) = om(\bar{x_{i2}}) = om(z_{i3}) = 0 \tag{19}$$

$$om(y_i) = om(z_{i1}) = 1 \tag{20}$$

For each clause $c_j \in C$, $1 \leq j \leq m$, with literals l_{j1}, l_{j2}, and l_{j3}, add the equation

$$(((g_j + f_{j1}) + f_{j2}) + f_{j3}) = h_j \tag{21}$$

where the quantity f_{jk} is v_i if l_{jk} is u_i, and $\bar{v_i}$ if l_{jk} is $\bar{u_i}$, for some $1 \leq i \leq n$. Add g_j and h_j to V. Add g_j to X, and define its order of magnitude as follows:

$$om(g_j) = 1 \tag{22}$$

Add the following equation to E:

$$h_1 * h_2 * \ldots * h_m = q \tag{23}$$

Let s be such that all the quantities in V, except z_i and z_{i3} ($1 \leq i \leq n$), are positive, and let the signs of z_i and z_{i3} be unknown. Let B be $3m - 1$.

That completes the reduction. Clearly, it can be done in polynomial time. We now show that any assignment of orders of magnitudes to the quantities of $\mathcal{I}_2$ that satisfies Equations 13–20, according to the rules in Figure 3, assigns the order of magnitude 1 to exactly one of v_i and $\bar{v}_i$, for each $1 \leq i \leq n$, and 0 to the other.

From rule 1 applied to Equation 13 we have:

$$om(x_{i1}) + om(x_{i2}) \leq om(v_i) \leq om(x_{i1}) + om(x_{i2}) + 1 \tag{24}$$

Substituting the orders of magnitudes of x_{i1} and x_{i2} (Equation 19) into the above equation, we have

$$0 \leq om(v_i) \leq 1 \tag{25}$$

In a similar way, rule 1 applied to Equation 14 implies that:

$$0 \leq om(\bar{v}_i) \leq 1 \tag{26}$$

Applying rule 1 to Equation 15 leads to:

$$om(v_i) + om(\bar{v}_i) \leq om(y_{i1}) \leq om(v_i) + om(\bar{v}_i) + 1 \tag{27}$$

Equation 16 implies that $om(y_i) \geq om(y_{i1})$. This fact follows from the observation that in rule 3:

$$om(q_1 + q_2) \geq om(q_1) \tag{28}$$

$$om(q_1 + q_2) \geq om(q_2) \tag{29}$$

under all three conditions. Since $om(y_i) = 1$ (Equation 20), it follows that

$$om(y_{i1}) \leq 1 \tag{30}$$

Hence, from Equations 27 and 30 it follows that:

$$om(v_i) + om(\bar{v}_i) \leq 1 \tag{31}$$

Now, since $om(z_{i1}) = 1$ (Equation 20), it follows from Equation 28 that

$$om(z_{i1} + z_{i2}) \geq 1 \tag{32}$$

Rule 1 applied to Equation 18 leads to:

$$om(z_{i1} + z_{i2}) + om(z_{i3}) \leq om(z_i) \tag{33}$$

Hence, from Equations 32 and 19, it follows that:

$$om(z_i) \geq 1 \tag{34}$$

Now, rule 4 implies that

$$om(q_1 - q_2) \leq maximum\{om(q_1), om(q_2)\} \tag{35}$$

Hence, from Equations 17, 25, 26, and 35, it follows that at least one of $om(v_i)$ and $om(\bar{v_i})$ must be 1. But since Equation 31 tells us that that their sum must be less than or equal to 1, it follows that exactly one of $om(v_i)$ and $om(\bar{v_i})$ must be 1, with the other being 0.

We now show that $\mathcal{I}_1$ has a satisfying truth assignment if and only if $\mathcal{I}_2$ is such that the maximum value of $om(q)$ is greater than or equal to B.

($\Rightarrow$) Let $\mathcal{I}_1$ have a satisfying truth assignment. We now assign order of magnitudes to the quantities of $\mathcal{I}_2$ such that Equations 13–23 are satisfied according to the rules of Figure 3. For $1 \leq i \leq n$, if u_i is true, then let $om(v_i)$ be 1 and $om(\bar{v_i})$ be 0; otherwise let $om(v_i)$ be 0 and $om(\bar{v_i})$ be 1. Since exactly one of $om(v_i)$ and $om(\bar{v_i})$ is 1 and the other is 0, this assignment of orders of magnitude will satisfy Equations 13–20.

Since the truth assignment satisfies every clause $c_j = \{l_{j1}, l_{j2}, l_{j3}\}, 1 \leq j \leq m$, it follows that at least one of l_{j1}, l_{j2}, or l_{j3} is true. Hence, at least one of f_{j1}, f_{j2}, or f_{j3} has an order of magnitude of 1. Since $om(g_j) = 1$ (Equation 22), it follows from Equation 21 and rule 3 that the maximum value of $om(h_j)$ is 2. Hence, from Equation 23 and rule 1, it follows that the maximum value of q is $3m - 1$ ($2m$ from each of the $om(h_j)$, and $m - 1$ from the product of m factors). Hence, the maximum value of $om(q)$ is greater than or equal to B.

($\Leftarrow$) Let us now assume that the maximum value of $om(q)$ is greater than or equal to B. Consider the assignment of order of magnitudes to quantities that supports $om(q)$ taking on its maximum value. For each variable u_i, $1 \leq i \leq n$, let u_i be true if and only the order of magnitude of quantity v_i is 1 in the above assignment. This gives us a well defined truth assignment since we have already shown that exactly one of $om(v_i)$ and $om(\bar{v_i})$ is 1. To show that this truth assignment satisfies very clause, we proceed as follows.

Since each $f_{jk}(1 \leq j \leq m, 1 \leq k \leq 3)$ is either v_i or $\bar{v_i}$, for some i, Equations 25 and 26 tell us that the maximum value of $om(f_{jk})$ is 1. Hence, using Equation 21 and rule 3, the maximum value of $om(h_j)$ can be 2. However, for the maximum value of $om(q)$ to be greater than or equal to $B(= 3m - 1)$, it follows that $om(h_j)$ must be 2. Hence, at least one of the f_{jk} must have an order of magnitude of 1. Hence, at least one of the l_{jk} will be true, and hence the truth assignment constructed above will satisfy each clause.

Hence, we have shown that $\mathcal{I}_1$ has a satisfying truth assignment if and only if $\mathcal{I}_2$ is such that the maximum value of $om(q)$ is greater than or equal to B. Hence, the ORDER OF MAGNITUDE REASONING problem is NP-hard.

B Derivation of error estimates

In this appendix we give a brief outline of the derivation of ϵ(Rule 3b). Outlines of the derivations of the other error estimates are similar and can be found in [12]. As discussed in Section 5, ϵ(Rule 3b) is given by the following:

$$\begin{aligned}\epsilon(\text{Rule 3b}) &= 1 - Prob\{b^{n_1} \leq Q < b^{n_1+1}\} \\ &= 1 - \int_{-\infty}^{\infty} \int_{b^{n_1}-q_1}^{b^{n_1+1}-q_1} f_{Q_1,Q_2}(q_1,q_2)dq_2dq_1\end{aligned}$$

Hence, from Assumption 1 (Equation 8) we get:

$$\epsilon(\text{Rule 3b}) = 1 - \int_{-\infty}^{\infty} \int_{b^{n_1}-q_1}^{b^{n_1+1}-q_1} f_{Q_1}(q_1)f_{Q_2}(q_2)dq_2dq_1$$

We now use Assumption 2 to split the above integral into two integrals, such that the integrand in both integrals is a non-zero constant throughout the region of integration:

$$\begin{aligned}\epsilon(\text{Rule 3b}) &= 1 - \int_{b^{n_1}}^{b^{n_1+1}-b^{n_2+1}} \int_{b^{n_2}}^{b^{n_2+1}} \frac{1}{b^{n_1+n_2}(b-1)^2}dq_2dq_1 \\ &\quad - \int_{b^{n_1+1}-b^{n_2+1}}^{b^{n_1+1}-b^{n_2}} \int_{b^{n_2}}^{b^{n_1+1}-q_1} \frac{1}{b^{n_1+n_2}(b-1)^2}dq_2dq_1 \\ &= 1 - \int_{b^{n_1}}^{b^{n_1+1}-b^{n_2+1}} \frac{b^{n_2+1}-b^{n_2}}{b^{n_1+n_2}(b-1)^2}dq_1 \\ &\quad - \int_{b^{n_1+1}-b^{n_2+1}}^{b^{n_1+1}-b^{n_2}} \frac{b^{n_1+1}-b^{n_2}-q_1}{b^{n_1+n_2}(b-1)^2}dq_1 \\ &= \frac{b+1}{2b^{n_1-n_2}(b-1)}\end{aligned}$$

References

[1] Scott W. Bennett. Approximation in mathematical domains. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 239–241, Los Altos, CA, August 1987. International Joint Conferences on Artificial Intelligence, Inc., Morgan Kaufmann Publishers, Inc.

[2] Wilbur B. Davenport, Jr. *Probability and Random Processes.* McGraw-Hill Book Company, 1970.

[3] Michael R. Garey and David S. Johnson. *Computers and Intractability.* W. H. Freeman and Company, 1979.

[4] Fredrick S. Hillier and Gerald J. Lieberman. *Introduction to Operations Reesearch.* Holden-Day, Inc., third edition, 1980.

[5] Yumi Iwasaki and Herbert A. Simon. Causality in device behavior. *Artificial Intelligence*, 29:3–32, 1986.

[6] Richard M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

[7] L. G. Khachian. A polynomial algorithm in linear programming. *Dokl. Akad. Nauk. SSSR*, 244:1093–1096 (in Russian), 1979. English translation in *Soviet Math. Dokl.* **20** (1979), 191–194.

[8] Benjamin Kuipers and Daniel Berleant. Using incomplete quantitative knowledge in qualitative reasoning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 324–329. American Association for Artificial Intelligence, 1988.

[9] M. Mavrovouniotis and G. Stephanopolous. Reasoning with orders of magnitude and approximate relations. In *Proceedings of the Sixth National Conference on Artificial Intelligence.* American Association for Artificial Intelligence, 1987.

[10] Ramon E. Moore. *Methods and Applications of Interval Analysis.* SIAM Studies in Applied Mathematics. SIAM, Philadelphia, 1979.

[11] Seshashayee S. Murthy. Qualitative reasoning at multiple resolutions. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 296–300. American Association for Artificial Intelligence, 1988.

[12] P. Pandurang Nayak. *Automated Modeling of Physical Systems.* PhD thesis, Stanford University, Department of Computer Science, Stanford, CA, 1992.

[13] P. Pandurang Nayak, Leo Joskowicz, and Sanjaya Addanki. Automated model selection using context-dependent behaviors. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 710–716. American Association for Artificial Intelligence, July 1992.

[14] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in Pascal: The Art of Scientific Computing.* Cambridge University Press, 1989.

[15] Olivier Raiman. Order of magnitude reasoning. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 100–104. American Association for Artificial Intelligence, 1986.

[16] Olivier Raiman. Order of magnitude reasoning. *Artificial Intelligence*, 51:11–38, 1991.

[17] Elisha Sacks. Hierarchical reasoning about inequalities. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 649–654. American Association for Artificial Intelligence, Morgan Kaufmann Publishers, Inc., July 1987.

[18] Reid Simmons. "Commonsense" arithmetic reasoning. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 118–124. American Association for Artificial Intelligence, 1986.

[19] Daniel S. Weld. Exaggeration. *Artificial Intelligence*, 43(2), 1990.